White Paper

The Powerful Combination of Linux* and the Intel® Itanium® Processor Family

Table of Contents

The Growth of Linux

In 1991 Linus Torvalds, a student at the University of Helsinki, began creating an open source operating system that could be run on an Intel386™ processor. He named the software Linux* and began distributing it free of charge over the Internet. Over the past decade, Linux has been gaining in popularity and functionality, thanks to thousands of developers loosely collaborating across the Internet. Linux continues to be developed under the GNU* General Public License. It has become a robust operating system used by people (primarily in businesses) mainly for front-end Internet infrastructure applications, such as Web servers, security servers, directory servers and firewalls. In addition to being free of license charges, one of the advantages of Linux over commercial versions of UNIX is that it supports a broader range of hardware and software. While each commercial version of UNIX targets a particular architecture, typically RISC, Linux runs on a variety of architectures, including Intel® architecture (both IA-32 and Intel® Itanium® architecture), Alpha,* PowerPC,* SPARC* and PA-RISC.* Linux also is capable of running the same types of applications and services that other UNIX systems run. Many server programs that are basic to UNIX have been ported to Linux as well. The Linux security architecture lends itself to a high degree of protection when configured appropriately. Furthermore, the fact that Linux is open source software encourages a continuous stream of new applications and development tools for the operating system. Linux offers other benefits of open source software as well, from lower risk to easier customization.

Running Linux on Itanium®-based Servers

As the number of Linux users and applications grow, so will the demand for more powerful servers to support them. Intel® Itanium® architecture is ideally suited to meet the challenge. Itanium architecture offers many features that provide a means of supplying more power on fewer machines. With the introduction of the Intel® Itanium® 2 processor, the second member of the Itanium processor family, the level of power has been increased. Intel estimates that software written for the Itanium processor will offer between 1.5 and 2 times performance on the Itanium 2 processor, without recompilation.

The Intel® Itanium® Processor Family and Linux* Project

The Intel® Itanium® processor and Linux* Project was launched by Intel, Hewlett-Packard, SGI, VA Linux and IBM in May 1999. The goal was to create a Linux port for the Itanium processor that would be optimized for Itanium architecture and available to the open source community in time for the Itanium processor launch. The Itanium processor and Linux Project is using the classic Linux development model, drawing from many sources in a cooperative effort to ensure delivery of the best possible code. Today, the Itanium processor and Linux Project team includes members from Caldera Systems, CERN, Hewlett-Packard, IBM, Intel, NEC, Red Hat, SGI, SuSE, and TurboLinux. The project bore fruit quickly following the May 29th, 2001 release of the Intel Itanium processor. Today, Linux releases for Itanium architecture are available from Red Hat, TurboLinux, Caldera, SuSe and MandrakeSoft.

Open Source Software Criteria

For software to be considered "open source," it must meet the following nine criteria (established by the Open Source Initiative):

1. **Free Distribution:** The software license may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.

2. **Access to Source Code:** The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost – preferably, free downloading via the Internet. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms, such as the output of a preprocessor or translator, are not allowed.

3. **Derived Works Allowed:** The license must allow for modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. **Integrity of Author's Source Code Must be Maintained:** The license may restrict source code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code, for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. **No Discrimination Against Persons or Groups:** The license must not discriminate against any person or group of persons.

6. **No Discrimination Against Use in Any Field of Endeavor:** The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. **Distribution of License:** The rights attached to the program must apply to all to whom the program is redistributed, without the need for execution of an additional license by those parties.

8. **License Must Not be Specific to a Product:** The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. **License Must Not Restrict Other Software Distributed with It:** The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

Unfortunately, because the term "open source" is descriptive, it cannot be protected as a trademark. As a result, the term is subject to misuse. Since the software development and user community needs a reliable way of knowing whether a piece of software conforms to the open source criteria listed above, a certification mark, "OSI Certified," has been registered for this purpose.

Benefits of Open Source Software

Open Source software has many benefits over commercial versions. Following are some of the key benefits:

Lower Risk

Having the source code gives customers control over the tools their business depends upon. When the developer of an open source product raises prices excessively, adds unacceptable licensing restrictions, or in any other way fails to please its customers, a different organization often uses the source code to start a new product that resolves the problems with the original developer. Customers also can choose to maintain the software themselves or to hire someone else to step in and better meet their needs.

Continuous Improvement

Open source software is becoming more popular in the commercial area, in part because of more effective development models, greater pride of authorship and massive independent peer review. Any person interested

in using open source software can review both the code and design and contribute to improving them. Some companies even go so far as to offer rewards for finding bugs in their open source software.

Transparency

Proprietary software has many "dark corners" in which bugs and other quirks can hide. Source code is crucial for debugging and understanding how a product works. In large software companies, few employees have access to source code, and the employees who do have access are rarely available directly to customers. Access to source code also is essential for finding and fixing security holes and for addressing other problems.

Optimum Features

Some open source products have been so effective that they have few viable commercial competitors. Following are just a few examples of open source software in heavy commercial use:

- Apache*, which runs Web servers.
- Perl*, the engine behind "live content" on the World Wide Web.
- BIND*, the software that provides the DNS (domain name service) for the Internet.
- sendmail*, a widely used e-mail transport software on the Internet.

Ability to Customize

The open source model gives customers a much greater ability to tailor software to fit their business needs. With the open source model, large organizations can easily make and deploy customizations on a large scale, reaping substantial cost and labor savings in the process. In addition, custom bug fixes and enhancements are commonly contributed to standard open source packages and thus become maintained – an option that is rarely available with traditional commercial software.

Favorable Licensing and Lower Cost

Open source software comes with far more flexible licensing terms than does proprietary software. This can substantially reduce the time and cost required for further installations, especially in organizations with a time- or labor-intensive procurement process. It also affords the customer greater flexibility in how the software is deployed or redistributed, and can save companies a great deal of effort in the details of license tracking and enforcement.

History of UNIX

UNIX was developed in the 1970s at Bell Labs by Ken Thompson and Dennis Ritchie as a multitasking operating system designed to run on minicomputers. The popularity of the operating system grew rapidly as it was supplied to universities, complete with source code, at no charge. UNIX became even more popular as college graduates using UNIX formed new companies (e.g.,Sun Microsystems) and as graduates were hired by companies, such as IBM, where their influence on the direction of new operating systems development was felt. Today, many commercial versions of UNIX are available, with many major hardware vendors offering their own versions.

Linux* Compared to UNIX

Several popular implementations of UNIX for the personal computer have been developed. A number of vendors have taken advantage of Intel® architecture and the way it lends itself to the UNIX design, including SCO and BSDI. Even Sun Microsystems has a version of Solaris* ported to x86. Many commercial implementations of UNIX are quite similar and are POSIX-compliant. (For an explanation of POSIX, see www.pasc.org) Almost all versions support the same software, programming environments and networking features. Linux follows this same paradigm for compatibility and functionality. In fact, many utilities found on commercial versions of UNIX have been ported to Linux. However, major differences exist between Linux and the commercial versions of UNIX. Some of the differences explain the growing popularity of Linux. One important difference is that Linux will run on any Intel®-based server, including the Intel® Itanium® processor. In fact, Linux originally was designed for Intel architecture. Today, more than 90 percent of Linux installations are on Intel®-based systems. Because Linux works on Intel architecture, it supports a wide variety of hardware, which in turn supports the majority of sound, graphics, network and SCSI card available. In addition, under the Open Source model on which the Linux operating system is based, anyone with enough motivation to write a driver for a particular piece of hardware may do so, adding to the list of hardware that Linux can support. Another advantage of Linux is price. Linux software is available via the Internet at no charge. Since the Linux operating system is Open Source software, users also may copy it from anyone else who has the software, without concern for licensing fees. There also are several low-cost means of obtaining the soft

ware whereby Linux may be bundled with other value-added items, such as additional software, manuals and technical support. Finally, an on-going effort called the Linux Documentation Project (LDP) offers free on-line access at **www.tldp.org** for documentation of the GNU/Linux.

Server Programs Available on Linux

Many server programs that are considered part of the basic functionality of UNIX have been ported to Linux as well. These programs include the following:

- **Domain Name Services (DNS):** A means of resolving computer names to their IP address for purposes of establishing IP connectivity.
- **Network File Sharing (NFS):** Allows files to be shared directly with a network of machines.
- **Network Information Services (NIS):** Allows a system to obtain information automatically on user accounts, groups, file system mount points and other system databases from servers on the network.
- **Samba:** A service that allows Windows* users to access Linux files and printers.
- **sendmail:** A service to handle sending and receiving e-mail. Many third-party vendors of server software also are in the process of porting their applications to Linux. Some of the more popular applications include streaming media services, Internet applications servers and a variety of network security services. The users of these third-party services range from universities to e-Business firms.

Linux* Security Architecture

System and network security is increasingly becoming a concern as more frequent and destructive attacks are being reported. The Linux* security architecture lends itself to a high degree of protection when each of its components is evaluated and configured appropriately for the risk of the environment. A system may be subjected to many types of attacks, from brazen theft of hardware to packet sniffing, password guessing, DNS spoofing, denial of service, viruses and Trojan horses. Following is an overview of system security issues and some of the ways in which Linux is designed to counter attacks.

- **Physical security:** This issue applies to any architecture. Since nearly all computer systems are vulnerable to onsite attack, the machines should be protected behind locked doors. Only those people having a genuine need to gain physical access should be permitted in the machine room. It may be appropriate in high-security areas to limit access on an individual basis by day of the week and hour of the day as well.
- **User accounts:** The Linux architecture is structured so that all administrative power is vested in a single account called root. This account enables an organization to control access to all resources of the system on an individual, group or global basis. A user is granted access to the system through a user ID/password combination. Obviously, this is the first line of defense against unauthorized access. A protected, hard-to-guess yet easy-to-remember password that is changed frequently is the best first-line defense against unauthorized system access. After a user logs into the system, the user is subject to the discretionary access controls placed on files, printers and other system resources by the system administrator (root). For instance, users may be grouped to allow read-only, read-write, or no access to system resources. In addition, Linux also provides the ability to selectively allow users or machines to connect to one another.
- **Firewall:** This is a device that prevents outsiders from accessing an internal network. It is typically a router, a stand-alone computer running packet filtering or proxy software, or a proprietary hardware device called a firewall-in-a-box, which filters and proxies.
- **Monitoring:** After properly configuring a system, the next best tool for protecting the system against misuse is diligent logging and viewing the system logs for suspicious activity. Linux is capable of logging all system and kernel messages, each network connection, which files are accessed by remote users, and every command a user enters. These logs are invaluable when investigating network intrusions.
- **Intrusion detection:** Between the tools that ship with Linux and the add-ons available from the Internet, an advanced intrusion capability can be established. Linux can log intrusion attempts and page the security administrator when such attacks occur. The operating system can be configured such that predefined actions are taken when attacks meet certain criteria (such as mimicking an operating system other than Linux to throw off the attacker).

- **Encryption:** Normally, when data are transmitted over the Internet, they traverse many gateways. Along this journey, the data are vulnerable to electronic eavesdropping. Various add-on Linux utilities make it possible to encrypt or scramble data so that if they are captured, they are unintelligible to the eavesdropper. For example, the credit card information entered during an online transaction is encrypted before it leaves the internal network and stays that way until the commerce server decrypts it. This shields the data from attack and provides secure electronic commerce.

- **Certificates:** These are electronic forms of validation that the supplier is a reputable e-Commerce trading partner. A trusted third-party issues certificates and verifies their authenticity. These certificates are becoming increasingly necessary to participate in the e-Commerce trading world.

- **Malicious code protection:** Viruses and Trojan horses are two ways in which malicious code can cause system damage and clog system and network resources. It is important to preserve a snapshot of a system immediately after installation. This allows for comparing system objects against themselves later in time to ensure the system has not changed. It also provides a reference point in the event a system is damaged by a malicious code attack. Several software packages are available to provide this type of reconciliation information, such as TripWire.*

  Virus scanning software is specifically designed to look within a system's files for patterns associated with known viruses. The software will, upon locating an infected file, report the discovery and proceed to disinfect the file.

- **Disaster recovery:** The prospect of rebuilding a system after the data have been destroyed is not a pleasant one, but failing to plan for that possibility will almost assuredly have dire consequences. Disasters can come in many forms, such as natural disasters (storm, flood, fire), human error (inadvertent file deletion), hardware failure (disk head crash), or software bug (program corrupts file). Preparing for a disaster can reduce stress and down time if one should occur.

  Some of the more important steps to prepare for recovery are to:

  - install a reconciliation package (such as TripWire)
  - keep records of software that is installed
  - perform regular system backups with verification (read after write) to a removable media
  - keep the system backups in a safe place (environmentally controlled, fire resistant and remote enough so that it is not likely to be affected by the same disaster that affected the system to be rebuilt).

Linux* Development Tools

Linux* includes everything necessary to develop software, including standard libraries, programming tools, compilers and debuggers that would be found on any UNIX system. Several tools have already been ported to Itanium architecture. In addition, several projects are underway to port other open source tools to Linux. A partial listing of tools that have, or are in the process of being, ported include the following:

Applications Tuning

- **VPROF:** System event counter monitoring. Provides hotspot execution profiling for application programs.
- **RABBIT:** Micro-architecture event sampling in time domain. Uses on-chip processor performance counters.

System Monitoring

- **TOP:** Provides an ongoing look at processor activity in real time; obtains system load information, shared memory usage, load averages, etc.; also displays a listing of the most processor-intensive tasks on the system.
- **VMSTAT:** Provides information about memory and swap areas, number of page swaps, processor time, etc.
- **PROCMETER:** A performance metering/logging program. Monitors system resources and presents it in a graphical window. The program can monitor a local or remote system.

Additional Tools for Linux* Development

In addition to the port of open source tools mentioned above, there are a number of other tools available from Intel and third parties to assist in Linux* development for the Itanium processor family, including:

Compilers:

- Intel® C++ Compiler 6.0 for Linux
- Intel® Fortran Compiler for Linux

Applications Tuning:

- Intel® Math Kernel Library 5.2 for Linux
- Intel® VTune™ Performance Analyzer

Features of the Intel® Itanium® Architecture

Intel® Itanium® architecture provides the capability to supply more power on fewer machines. Included in these new features are the following:

- **Instruction level parallelism (ILP):** The compilers for the Itanium processor are responsible for generating groups of independent instructions to be submitted to the execution units. Historically, ILP has been performed in hardware; placing this functionality in the compiler frees valuable silicon real estate.
- **Massive register set:** There are 128 integer registers, 128 floating point registers, 64 predicate registers, 8 branch registers and various other registers used by the operating system to improve performance as well.
- **Register stack engine:** The general-purpose registers are used to pass parameters to and from subroutines to hide memory latency. The register stack engine provides a means of transparently saving and retrieving values from the register stack to accommodate stack overflow while continuing to minimize the latency.
- **Predication:** The instruction set allows for each instruction to be executed conditionally based on some previously determined criterion. This allows branches to be removed from code and improves the runtime performance.
- **Software pipelining:** Traditional architectures perform loop unrolling, which results in code expansion and also increases cache misses. The Itanium architecture uses predication and register renaming to overlapping execution of multiple loop instances. This further increases the runtime performance.
- **Branch hints:** The compiler and processor registers provide information to further reduce the number of branches encountered at runtime for additional performance gains.
- **Control and data speculation:** The processor will load data and target addresses into registers before they are actually needed to further hide memory latency.
- **Floating-point architecture:** New floating-point instructions enhance floating-point performance and enable the Itanium processor to achieve world-class multimedia performance.
- **Task level parallelism (TLP):** Multiple processor configurations of Itanium architecture allow the operating system to assign those extra resources to execute multiple threads of a program in parallel, improving runtime performance.

Linux* and Intel® Itanium® Architecture in High Performance Technical Computing

In virtually every field of science and engineering, research and development teams are searching for new quantitative precision, and trying to model increasingly complex systems. Not surprisingly, this is one of the first market segments to begin embracing the power and affordability of the combination of Linux and Itanium®-based solutions. One example is the supercomputer currently under development by the National Science Foundation. This distributed system will include both Intel® Itanium® processors and Intel® Itanium® 2 processors, with a combined total of over 3,300 processors. Platforms will be hosted at four separate super-computing institutions, and will be linked together via dedicated high-speed interconnects. The total system will be capable of 13.6 trillion calculations per second, and able to work with 450 terabytes of data.

Another example is the supercomputer being developed for the Department of Energy's Pacific Northwest National Laboratory (PNNL). With 1,400 Intel Itanium 2 processors, 1.8 terabytes of memory and 170 terabytes of disk space, this new system will be able to perform up to 8.3 trillion floating point operations per second. When completed, it will be one of the world's most powerful Linux-based supercomputers, enabling PNNL researchers to solve complex calculations 30 times faster than they could with their previous system.

Linux* and Intel® Itanium® Architecture in Enterprise Computing

Linux on Intel® Itanium® architecture is not restricted to HPTC, however. Many vendors of enterprise software also recognize the advantages of Linux, and are working to make available Linux versions of leading applications, optimized for Intel® architecture. Examples include both Oracle and IBM, who plan to make available Linux versions of their powerful RDBMS systems.

Web Resources

- **SGI:** C/C++/Fortran compilers, GNU C library (glibc): **oss.sgi.com/projects/Pro64**
- **IBM:** Java* 1.3 for Linux: **www6.software.ibm.com/dl/dklx130/dklx130-p**
- **Python* software: www.python.org/download**
- **Perl software: www.perl.com/pub/language/info/software.html**
- **PHP downloads: www.php.net/downloads.php**
- **Tcl/TK*: www.scriptics.com/software/tcltk/download83.html**

References

- ***Intel® Itanium® Architecture ISV Technical Presentation***
- ***Intel® Itanium® Processor Porting ISV Technical Presentation***
- ***Open Source 101* presentation:** IACG/WPG IM ISV-IHV Enabling Francisco Azuola, April 2000
- ***Linux* Performance and Methodologies* presentation:** Linux Performance Group, MSL/IPL, Santa Clara Vik Kumar, March 2000
- ***Running Linux*,** third edition, Matt Welsh, Mathias Kalle Dalheimer and Lar Kaufman O'Reilly, August 1999
- ***Maximum Linux* Security,*** Anonymous, Sams, September 1999
- ***Operating Systems,*** second edition, H. M. Deitel, Addison Wesley, 1990
- ***IA-64 Linux Kernel: Design and Implementation,*** Mosberger & Eranian

0602/HB/SE/PDF

283573-002
SEG_139